LOGGING CACHE INFLUXES BY REQUEST TO A HIGHER-LEVEL CACHE

CROSS REFERENCE TO RELATED APPLICATION

This application is a continuation-in-part of U.S. patent application Ser. No. 15/904,072, filed Feb. 23, 2018, and titled "TRACE RECORDING BY LOGGING INFLUXES TO A LOWER-LAYER CACHE BASED ON ENTRIES IN AN UPPER-LAYER CACHE," the entire contents of which are incorporated by reference herein in their entirety.

BACKGROUND

[0001] When writing code during the development of software applications, developers commonly spend a significant amount of time"debugging" the code to find runtime and other source code errors. In doing so, developers may take several approaches to reproduce and localize a source code bug, such as observing the behavior of a program based on different inputs, inserting debugging code (e.g., to print variable values, to track branches of execution, etc.), temporarily removing code portions, etc. Tracking down runtime errors to pinpoint code bugs can occupy a significant portion of application development time.

[0002] Many types of debugging applications ("debuggers") have been developed in order to assist developers with the code debugging process. These tools offer developers the ability to trace, visualize, and alter the execution of computer code. For example, debuggers may visualize the execution of code instructions, may present code variable values at various times during code execution, may enable developers to alter code execution paths, and/or may enable developers to set"breakpoints" and/or"watchpoints" on code elements of interest (which, when reached during execution, causes execution of the code to be suspended), among other things.

[0003] An emerging form of debugging applications enable "time travel," "reverse," or "historic" debugging. With "time travel" debugging, execution of a program (e.g., executable entities such as threads) is recorded/traced by a trace application into one or more trace files. These trace file(s) can then be used to replay execution of the program later, for both forward and backward analysis. For example, "time travel" debuggers can enable a developer to set forward breakpoints/watchpoints (l ike conventional debuggers) as well as reverse breakpoints/watchpoints.

[0004] Several considerations can be taken into account when recording trace files. Most prominently, there is an inherent tradeoff between the robustness of the trace data recorded and the overheads incurred by tracing a program. These tradeoffs are manifest primarily in trace file size and performance impacts on execution of the traced program. Moreover, since tracing might be accomplished with hardware assistance (or entirely in software), there may also be hardware design and other hardware cost considerations.

BRIEF SUMMARY

[0005] Embodiments described herein are directed to mechanisms for creating bit-accurate "time travel" trace recordings using hardware assistance by a processor. These

What is claimed : ~~CLAIMS~~

1. A microprocessor, comprising:

a plurality of processing units;

a plurality of caches arranged into a plurality of cache layers, the plurality of caches including a plurality of first caches within a first cache layer and one or more second caches within a second cache layer, a particular second cache in the second cache layer serving as a backing store for at least a particular first cache in the first cache layer; and

control logic that configures at least the particular second cache to perform at least the following:

receive, from the particular first cache, a logging request referencing a particular memory address; and

based on the request, determine whether a cache line corresponding to the memory address is present in the particular second cache, and

when the cache line is not present in the particular second cache, perform one of:

cause the cache line to be logged when there does not exist a third cache that participates in logging and that serves as a backing store for at least the particular second cache, or

forward the request to the third cache when the third cache does exist; or

when the cache line is present in the particular second cache, perform at least one of:

cause the cache line to be logged when the cache line is (i) not determined by the particular second cache to be logged, or (ii) determined by the particular second cache to be logged but the particular second cache has not determined that the first cache is aware of a current value stored in the cache line of the particular second cache; or

determine that cache line need not be logged when (i) the cache line is determined by the particular second cache to be logged, and (ii) it is determined that the first cache is aware of the current value stored in the cache line of the particular second cache.

2. The microprocessor as recited in claim 1, wherein causing the cache line to be logged comprises:

logging the cache line within a trace buffer; and

marking the cache line as being logged within the particular second cache.

3. The microprocessor as recited in claim 1, wherein causing the cache line to be logged comprises one of:

instructing the first cache to log a value of the particular memory address directly; or

instructing the first cache to log a reference to a prior log entry for the particular memory address.

4. The microprocessor as recited in claim 1, wherein causing the cache line to be logged when the cache line is not determined by the particular second cache to be logged comprises:

determining that the third cache layer exists, and

notifying the third cache that the cache line has been logged by value by the particular second cache.

5 The microprocessor as recited in claim 1, wherein the first cache layer comprises an Ll cache layer, and wherein the second cache layer comprises an L2 cache layer or an L3 cache layer.

6. The microprocessor as recited in claim 1, the control logic also configuring at least the particular second cache to:

receive a message from the first cache indicating that another cache line in the first cache that also corresponds to the memory address is marked as not logged within the first cache; and

based on the message, mark the cache line as not logged within the particular second cache.

7. The microprocessor as recited in claim 6, wherein the other cache line is marked as not logged within the first cache based on the other cache line being written to by a processing unit corresponding to the first cache while logging is disabled for the processing unit.

8. The microprocessor as recited in claim 1, the control logic also configuring at least the particular second cache to:

receive a message from the first cache requesting the cache line for writing; and

send the cache line to the first cache, the cache line being marked as not logged based at least on logging being disabled for the first cache.

9. A method of an upper cache layer determining how to log an influx by a

lower cache layer, based on a logging request by the lower cache layer, the method being implemented at a computing device that includes (i) a plurality of processing units, (ii) a plurality of caches arranged into a plurality of cache layers, the plurality of caches including a plurality of first caches within a first cache layer and one or more second caches within a second cache layer, a particular second cache in the second cache layer serving as a backing store for at least a particular first cache in the first cache layer, the method comprising:

receiving, from the particular first cache, a logging request referencing a particular memory address; and

based on the request, determining whether a cache line corresponding to the memory address is present in the particular second cache, and

when the cache line is not present in the particular second cache, performing one of:

causing the cache line to be logged when there does not exist a third cache that participates in logging and that serves as a backing store for at least the particular second cache, or

forwarding the request to the third cache when the third cache does exist; or

when the cache line is present in the particular second cache, performing at least one of:

causing the cache line to be logged when the cache line is (i) not determined by the particular second cache to be logged, or (ii) determined by the particular second cache to be logged but the particular second cache has not determined that the first cache is aware of a current value stored in the cache line of the particular second cache; or

determining that cache line need not be logged when (i) the cache line is determined by the particular second cache to be logged, and (ii) it is determined that the first cache is aware of the current value stored in the cache line of the particular second cache.

10. The method of claim 9, wherein causing the cache line to be logged comprises:

logging the cache line within a trace buffer; and

marking the cache line as being logged within the particular second cache.

11. The method of claim 9, wherein causing the cache line to be logged comprises one of:

instructing the first cache to log a value of the particular memory address directly; or

instructing the first cache to log a reference to a prior log entry for the particular memory address.

12. The method of claim 9, wherein causing the cache line to be logged when the cache line is not determined by the particular second cache to be logged comprises:

determining that the third cache exists, and

causing the cache line to be logged by reference based on knowledge of the third cache.

13. The method of claim 9, wherein causing the cache line to be logged when the cache line is not determined by the particular second cache to be logged comprises:

determining that the third cache layer exists, and

notifying the third cache that the cache line has been logged by value by the particular second cache.

14. The method of claim 9, wherein the first cache layer comprises an Ll cache layer, and wherein the second cache layer comprises an L2 cache layer or an L3 cache layer.

15. The method of claim 9, the control logic also configuring at least the particular second cache to:

receive a message from the first cache indicating that another cache line in the first cache that also corresponds to the memory address is marked as not logged within the first cache; and

based on the message, mark the cache line as not logged within the particular second cache.

16. The method of claim 15, wherein the other cache line is marked as not logged within the first cache based on the other cache line being written to by a processing unit corresponding to the first cache while logging is disabled for the processing unit.

17. The method of claim 9, the control logic also configuring at least the particular second cache to :

receive a message from the first cache requesting the cache line for writing; and

send the cache line to the first cache, the cache line being marked as not logged based at least on logging being disabled for the first cache.

18. A microprocessor, comprising :

a plurality of processing units;

a plurality of caches arranged into a plurality of cache layers, the plurality of caches including a plurality of first caches within a first cache layer and one or more second caches within a second cache layer, the second cache layer serving as a backing store for at least a particular first cache in the first cache layer that corresponds to a particular processing unit of the plurality of processing unit; and

control logic that configures at least the particular first cache to perform at least the following :

detect a write to a cache line in the first cache that has a logging status that has been set;

based on detecting the write, determine whether logging is enabled for the particular processing unit, and based at least on logging being disabled for the particular processing unit, clear the logging status for the cache line and notify at least one of the one or more second caches that its logging status for the cache line should be cleared, or

based at least on logging being enabled for the particular processing unit, retain the logging status for the cache line.

19. The microprocessor of claim 18, wherein the particular processing unit has the cache line in an owned state, and wherein the control logic also configures at least the particular first cache to notify at least one of the plurality of first caches that its logging status for the cache line should be cleared.

20. The microprocessor of claim 18, wherein the particular first cache to notifies the at least one of the plurality of first caches that its logging status for the cache line should be cleared based on a request received from the at least one of the plurality of first caches.